A Complexity Approach to Tree Algebras: the Polynomial Case

Arthur Jaquard

joint work with Thomas Colcombet

Université de Paris, CNRS, IRIF, F-75006, Paris, France

September 17, Highlights 2021

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Université de Paris

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Objects

$$\begin{array}{c} a \\ / \quad \backslash \\ b \quad\; c \end{array} \in T_\emptyset \qquad \begin{array}{c} a \\ / \quad \backslash \\ x \quad\; x \end{array} \in T_{\{x\}}$$

$$\begin{array}{c} a \\ / \quad \backslash \\ x \quad\; y \end{array} \in T_{\{x,y\}}$$

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccc} & a & \\ / & & \backslash \\ \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} & & y \end{array}$$

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}}$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccc} & & a & & \\ & / & & \backslash & \\ & a & & & y \\ / & & \backslash & & \\ b & & c & & \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \overset{\sigma}{\mapsto} \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array}$$

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Objects

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} \in T_\emptyset \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array} \in T_{\{x\}} \qquad \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \cdot_x \begin{array}{ccc} & a & \\ / & & \backslash \\ b & & c \end{array} = \begin{array}{ccccc} & & a & & \\ & / & & \backslash & \\ & a & & & y \\ / & & \backslash & & \\ b & & c & & \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{ccc} & a & \\ / & & \backslash \\ x & & y \end{array} \overset{\sigma}{\mapsto} \begin{array}{ccc} & a & \\ / & & \backslash \\ x & & x \end{array}$$

A tree algebra $\mathcal{A} = (A_X)_{X \text{ finite}}$ is finite if all the sorts A_X are finite.

Infinitely sorted tree algebras

The free tree algebra has as carrier $(T_X)_{X \text{ finite}}$ where the X's are finite sets of variables.

$$T_X = \{\text{trees in which all the variables of } X \text{ appear on the leaves}\}$$

Objects

$$\begin{array}{c} a \\ / \ \ \backslash \\ b \quad\ c \end{array} \in T_\emptyset \qquad \begin{array}{c} a \\ / \ \ \backslash \\ x \quad\ x \end{array} \in T_{\{x\}}$$

$$\begin{array}{c} a \\ / \ \ \backslash \\ x \quad\ y \end{array} \in T_{\{x,y\}}$$

Substitution

$$\begin{array}{c} a \\ / \ \ \backslash \\ x \quad\ y \end{array} \cdot_x \begin{array}{c} a \\ / \ \ \backslash \\ b \quad\ c \end{array} = \begin{array}{c} \quad a \\ \ \ / \ \ \backslash \\ a \quad\ y \\ / \ \ \backslash \quad \\ b \quad\ c \quad \end{array}$$

Renaming

$$\sigma(x) = \sigma(y) = x$$

$$\begin{array}{c} a \\ / \ \ \backslash \\ x \quad\ y \end{array} \overset{\sigma}{\mapsto} \begin{array}{c} a \\ / \ \ \backslash \\ x \quad\ x \end{array}$$

A tree algebra $\mathcal{A} = (A_X)_{X \text{ finite}}$ is finite if all the sorts A_X are finite.

Definition (Language recognized by an algebra)

A language L of finite trees over Σ is recognized by a finite tree algebra $\mathcal{A}$ if it is the inverse image of a subset of $A_\emptyset$ under a morphism from the free tree algebra to $\mathcal{A}$.

Languages and the size of the algebra

Definition (Language recognized by an algebra)

A language L of finite trees over Σ is recognized by a finite tree algebra $\mathcal{A}$ if it is the inverse image of a subset of $A_\emptyset$ under a morphism from the free tree algebra to $\mathcal{A}$.

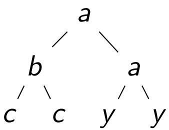
Example $L =$ trees without b's on the leftmost branch

Languages and the size of the algebra

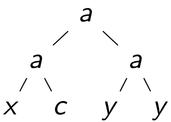
Definition (Language recognized by an algebra)

A language L of finite trees over Σ is recognized by a finite tree algebra $\mathcal{A}$ if it is the inverse image of a subset of $A_\emptyset$ under a morphism from the free tree algebra to $\mathcal{A}$.

Example $L =$ trees without b's on the leftmost branch



$\overset{\alpha}{\longmapsto} \{a, b, c\}$



$\overset{\alpha}{\longmapsto} (\{a\}, x),$

Languages and the size of the algebra

Definition (Language recognized by an algebra)

A language L of finite trees over Σ is recognized by a finite tree algebra $\mathcal{A}$ if it is the inverse image of a subset of $A_\emptyset$ under a morphism from the free tree algebra to $\mathcal{A}$.

Example $L =$ trees without b's on the leftmost branch

$$\begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & b & & & & a & \\ / & & \backslash & & / & & \backslash \\ c & & c & & y & & y \end{array} \overset{\alpha}{\longmapsto} \{a, b, c\} \qquad \begin{array}{ccccccc} & & & a & & & \\ & & / & & \backslash & & \\ & a & & & & a & \\ / & & \backslash & & / & & \backslash \\ x & & c & & y & & y \end{array} \overset{\alpha}{\longmapsto} (\{a\}, x),$$

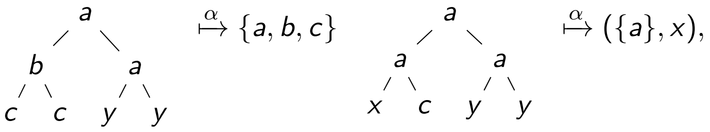
$$A_X = 2^\Sigma \uplus (2^\Sigma \times X) \qquad |A_X| = 2^{|\Sigma|} + 2^{|\Sigma|}|X| \text{ is linear in } |X|.$$

Languages and the size of the algebra

Definition (Language recognized by an algebra)

A language L of finite trees over Σ is recognized by a finite tree algebra $\mathcal{A}$ if it is the inverse image of a subset of $A_\emptyset$ under a morphism from the free tree algebra to $\mathcal{A}$.

Example $L =$ trees without b's on the leftmost branch



$A_X = 2^\Sigma \uplus (2^\Sigma \times X)$ $\qquad |A_X| = 2^{|\Sigma|} + 2^{|\Sigma|}|X|$ is linear in $|X|$.

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_\mathcal{A}(|X|) = |A_X|$.

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_\mathcal{A}(|X|) = |A_X|$.

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_{\mathcal{A}}(|X|) = |A_X|$.

2-EXP

EXP

Polynomial

Bounded

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_{\mathcal{A}}(|X|) = |A_X|$.

2-EXP

EXP

Polynomial

Bounded

Regular languages

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_\mathcal{A}(|X|) = |A_X|$.

2-EXP

Regular languages

EXP

Polynomial

Bounded

The objective is to identify new classes of languages and to gain a better understanding of tree algebras.

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_{\mathcal{A}}(|X|) = |A_X|$.

2-EXP	Regular languages
EXP	
Polynomial	
Bounded	[Colcombet & J. 2021]

The objective is to identify new classes of languages and to gain a better understanding of tree algebras.

Complexity

Definition (Complexity)

Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_{\mathcal{A}}(|X|) = |A_X|$.

2-EXP	Regular languages
EXP	
Polynomial	[This talk]
Bounded	[Colcombet & J. 2021]

The objective is to identify new classes of languages and to gain a better understanding of tree algebras.

Complexity

Definition (Complexity)

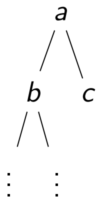
Given a finite FT_Σ-algebra $\mathcal{A}$ with carrier $(A_X)_{X \text{ finite}}$ its complexity map is $c_{\mathcal{A}}(|X|) = |A_X|$.

2-EXP	Regular languages
EXP	[Future work]
Polynomial	[This talk]
Bounded	[Colcombet & J. 2021]

The objective is to identify new classes of languages and to gain a better understanding of tree algebras.

The corresponding automaton model

How to build the following tree ?



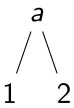
The corresponding automaton model

How to build the following tree ?

0

The corresponding automaton model

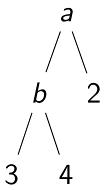
How to build the following tree ?



$$[0 \leftarrow a(1, 2)]$$

The corresponding automaton model

How to build the following tree ?

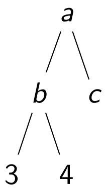


$$[0 \leftarrow a(1,2)]$$

$$[1 \leftarrow b(3,4)]$$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

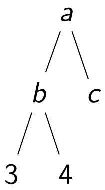
$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

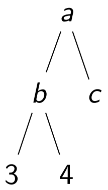
$[2 \leftarrow c]$

...

This is a word over an orbit-finite alphabet

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

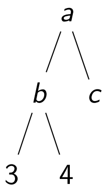
...

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree ($t(u) = t(v)$)

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

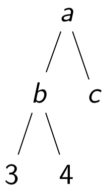
This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree ($t(u) = t(v)$)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v

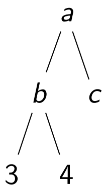
$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree ($t(u) = t(v)$)

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$
$[1 \leftarrow b(3,4)]$
$[2 \leftarrow c]$
...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

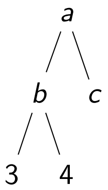
$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

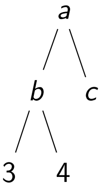
This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

Example: no b on the leftmost branch

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree ($t(u) = t(v)$)

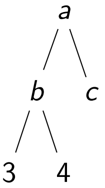
Example: no b on the leftmost branch

0

$(\emptyset, 0)$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$
$[1 \leftarrow b(3,4)]$
$[2 \leftarrow c]$
...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

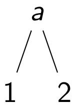
- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

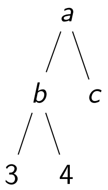
Example: no b on the leftmost branch



$(\{a\}, 1)$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$
$[1 \leftarrow b(3,4)]$
$[2 \leftarrow c]$
...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

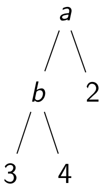
- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

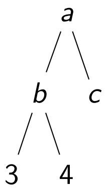
Example: no b on the leftmost branch



$(\{a, b\}, 3)$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$
$[1 \leftarrow b(3,4)]$
$[2 \leftarrow c]$
...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

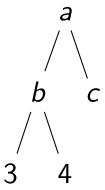
- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

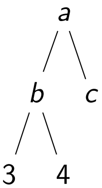
Example: no b on the leftmost branch



$(\{a, b\}, 3)$

The corresponding automaton model

How to build the following tree ?



$[0 \leftarrow a(1,2)]$

$[1 \leftarrow b(3,4)]$

$[2 \leftarrow c]$

...

Finite deterministic automaton $\mathcal{A}$ with k registers that store integers

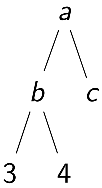
- If $t(u) = t(v)$ then $\mathcal{A}$ accepts u iff $\mathcal{A}$ accepts v
- Minor syntactic constraints (see poster for details)

$\mathcal{A}$ accepts $t(u)$ if it accepts u

This is a word over an orbit-finite alphabet

Note that two words u and v can denote the same tree $(t(u) = t(v))$

Example: no b on the leftmost branch



$(\{a, b\}, 3)$

Same expressive power as algebras of polynomial complexity.

Theorem on permutation groups

Theorem 5.2A. *Let $A := Alt(\Omega)$ where $n := |\Omega| \geq 5$,, and let r be an integer with $1 \leq r \leq n/2$. Suppose that $G \leq A$ has index $|A : G| < \binom{n}{r}$. Then one of the following holds:*

(i) *for some $\Delta \subseteq \Omega$ with $|\Delta| < r$ we have $A_{(\Delta)} \leq G \leq A_{\{\Delta\}}$;*

(ii) *$n = 2m$ is even, G is imprimitive with two blocks of size m, and $|A : G| = \frac{1}{2}\binom{n}{m}$; or*

(iii) *one of six exceptional cases hold where:*

  (a) *G is imprimitive on Ω and $(n, r, |A : G|) = (6,3,15)$;*

  (b) *G is primitive on Ω and $(n, r, |A : G|, G) = (5,2,6,5{:}2)$, $(6,2,6,PSL_2(5))$, $(7,2,15,PSL_3(2))$, $(8,2,15,AGL_3(2))$, or $(9,4,120,P\Gamma L_2(8))$.*

[DIXON, MORTIMER, Permutation groups. Springer Science & Business Media, 1996]

Theorem on permutation groups

Theorem 5.2A. *Let $A := Alt(\Omega)$ where $n := |\Omega| \geq 5$,, and let r be an integer with $1 \leq r \leq n/2$. Suppose that $G \leq A$ has index $|A : G| < \binom{n}{r}$. Then one of the following holds:*

(i) *for some $\Delta \subseteq \Omega$ with $|\Delta| < r$ we have $A_{(\Delta)} \leq G \leq A_{\{\Delta\}}$;*

(ii) *$n = 2m$ is even, G is imprimitive with two blocks of size m, and $|A : G| = \frac{1}{2}\binom{n}{m}$; or*

(iii) *one of six exceptional cases hold where:*

  (a) *G is imprimitive on Ω and $(n, r, |A : G|) = (6,3,15)$;*

  (b) *G is primitive on Ω and $(n, r, |A : G|, G) = (5,2,6,5{:}2)$, $(6,2,6,PSL_2(5))$, $(7,2,15,PSL_3(2))$, $(8,2,15,AGL_3(2))$, or $(9,4,120,P\Gamma L_2(8))$.*

[DIXON, MORTIMER, Permutation groups. Springer Science & Business Media, 1996]

Corollary: existence of small pseudo-support

Fix $k \in \mathbb{N}$, and let **Sym**(n) act transitively on A with $|A| \leq n^k$. If n is large enough then, for every $a \in A$, there is some $\Delta \subseteq n$ with $|\Delta| \leq k$ such that

$$\mathbf{Alt}(n \setminus \Delta) \subseteq \mathbf{Stabilizer}(a)$$